Project proposal for

Integrating ZOO services with QGIS

during Google Summer of Code 2022

Index

Contact Details

Name - Sandeepan Dhoundiyal
Nickname - Sandy
Country - India
Email - dsandeepan995@gmail.com
Phone - (+91)9176283450
Public repository - https://github.com/sdhoundiyal
Linkedin - https://www.linkedin.com/in/sandeepan-dhoundiyal-988244ab/

Idea

OSGeo incubation project: ZOO-Project

Title: Integrating ZOO services with QGIS

Project Description:

The ZOO-Project is an open-source implementation of the OGC's (Open Geospatial Consortium) Web Processing Service (WPS) standard.

WPS Interface Standard provides rules for standardizing how inputs and outputs (requests and responses) for geospatial processing services (eg. polygon overlay). WPS also standardizes how a client can request the execution of a process, and how the output from the process is handled.

The ZOO-Project consists of four components:-

1. The ZOO-Kernel:- C implementation of a WPS compliant server, that allows management and chaining of individual WPS services.
2. ZOO-Services:- A collection of geospatial tools implemented according to the WPS standard. These tools include methods from open-source libraries such as from open-source libraries: GDAL, GRASS GIS, OrfeoToolbox, CGAL and SAGA GIS.
3. ZOO-API:- A server-side JavaScript API managing available WPS services
4. ZOO-Client:- A client-side Javascript API for communicating with a WPS server.

While a number of tools are provided as ready-to-use services, their scope is limited, and these services need to be greatly expanded. This can be done by integrating ZOO-Services with the QGIS processing framework, which is a processing environment that can be used to call native, and third-party, algorithms from QGIS. By integrating the QGIS processing framework with Zoo-Services, processes compatible with QGIS can easily be incorporated into the ZOO-Project as first-citizen services. The project would thus allow processing tools that are natively part of QGIS to be incorporated as ZOO-Services, as well as provide support for the addition of QGIS plugins in the future.

The project would involve implementing a python based bridge for integrating QGIS processing framework similar to the wps-grass-bridge for GRASS GIS and the Optional Orfeo Toolbox support for the Orfeo Toolbox.

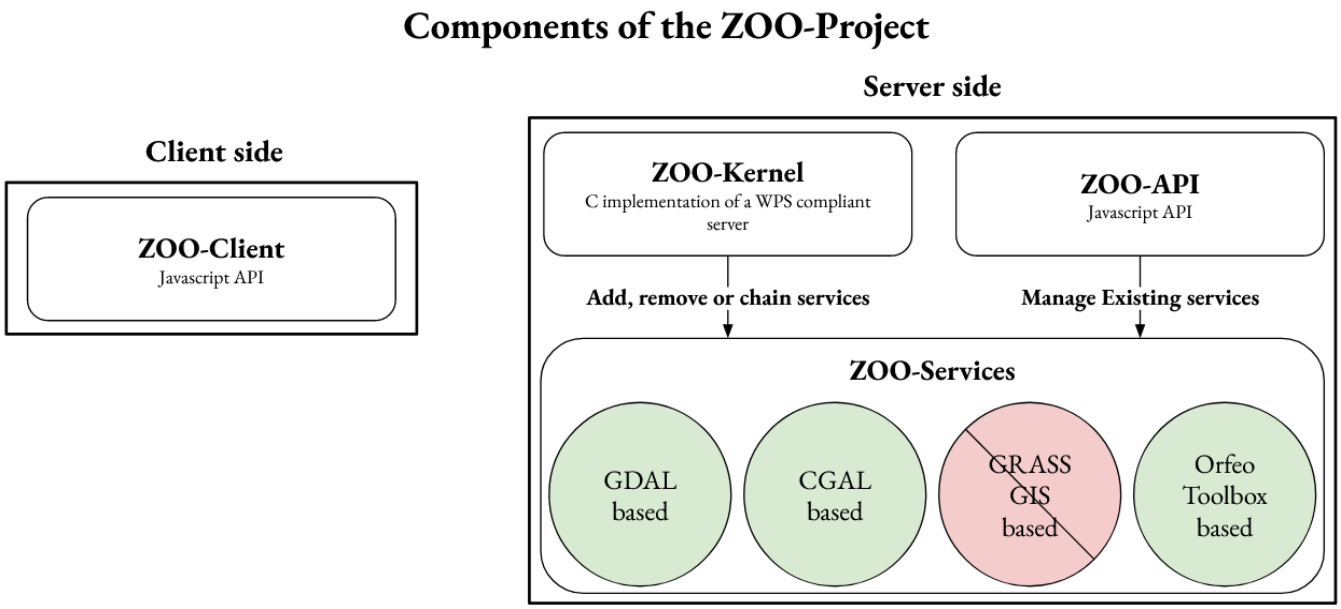


Fig. 1. State-of-the-art prior to GSoC

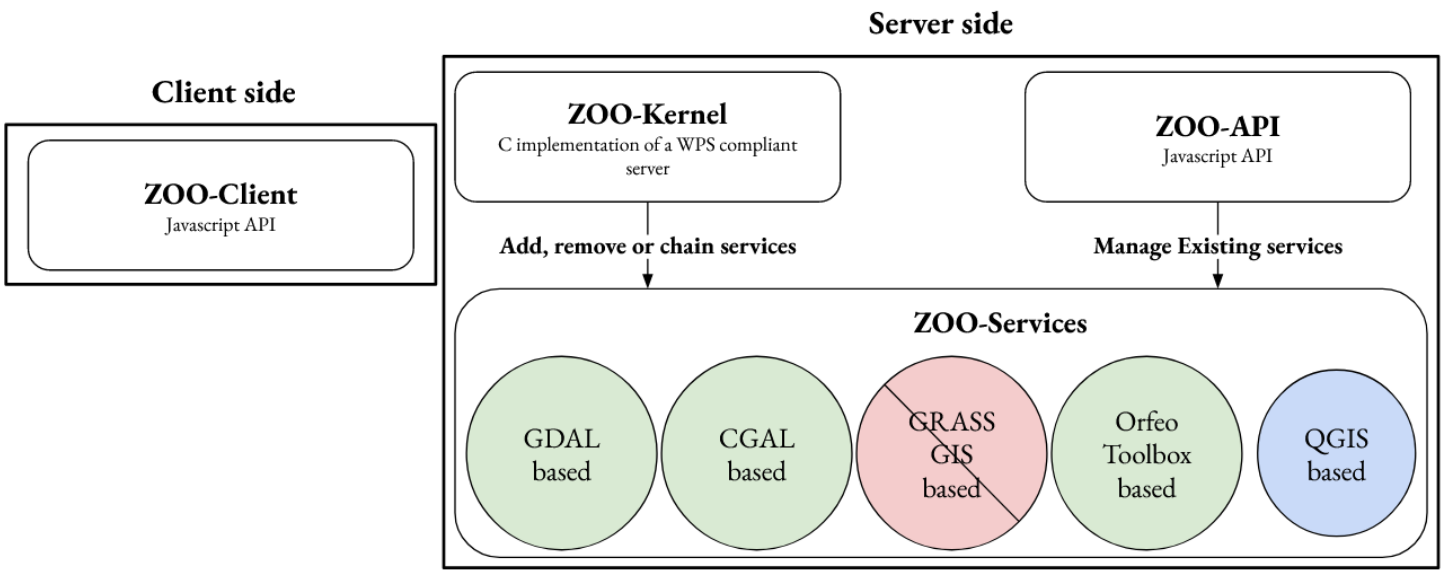


Fig. 2. State of the ZOO-Project after GSoC

State of the project before GSoC:

The ZOO-Project provides ready-to-use services implemented in C and Python. These services are based on open source libraries and are compliant with OGC standards, with minor modifications in their source code.

These services are:

Based on GDAL:-

GDAL (Geospatial Data Abstraction Library) is an open-source library for handling basic raster and vector geospatial data formats, translating between them, and performing basic geospatial processing. Documentation for the GDAL implementation of these services and other methods can be found here.

Table 1: GDAL based ZOO-services

Service	**Description**
Gdal_Contour	Builds vector contour lines from a raster elevation model
Gdal_Grid	Creates regular raster grid front the scattered data read from an OGR data source
Gdal_Dem	Tools for the analysis of raster elevation models
Gdal_Ndvi	Calculates Normalized Difference Vegetation Index on a raster file.
Gdal_profile	Fetches the elevation profile of a raster DEM in the form of XYZ values along a straight line
Gdal_translate	Converts raster data between different formats
Gdal_Warp	Mosaics, reprojects or warps raster images
Ogr2Ogr	Converts vector data between different formats
Base-vect-ops	Tools for spatial analysis of single and multiple vector geometries

Based on CGAL:-

CGAL (Computational Geometry Algorithms Library) is a C++ implementation of a collection of geometric algorithms, addressing needs in areas such as GIS (Geographic Information Systems), CAD (Computer-Aided Design), molecular biology, medical imaging, computer graphics, and robotics.

Table 2: CGAL based ZOO-services

Service	**Description**
Cgal_Delaunay	Computes edges of Delaunay triangulation given a set of data points. Documentation for the CGAL implementation can be found here.
Cgal_Voronoi	Computes edges of a Voronoi Polygon given a set of data points. Documentation for the CGAL implementation can be found here.

Based on GRASS GIS:-

GRASS(Geographic Resources Analysis Support System) GIS is a C++ based platform for processing, modelling, visualizing and managing geospatial data. In the past the wps-grass-bridge library of the ZOO-Project allowed GRASS GIS 7 modules to be used as ZOO-Services. However, programmes introduced in subsequent releases of GRASS GIS are not supported as ZOO-Services. Documentation for GRASS 8 can be found here.

Based on Orfeo Toolbox:-

The OrfeoToolbox is a collection of tools for processing remote sensing data. Use-cases include ortho-rectification, pan-sharpening, classification, SAR processing. Similar to GRASS GIS, Orfeo Toolbox Applications can be used as

ZOO-Services without any modification using the Orfeo Toolbox support. Documentation for the Orfeo Toolbox applications can be found here.

Benefits to the community/Additions the project will bring to the software:

The project will make a wide variety of services natively available through the ZOO-Project. These services would comply with the WPS standard, and also with the OGC API processes.

QGIS and the ZOO-Project are both important standard driven software, which would be brought together by this project. This would allow complex processes to be made available as ready-to-use interoperable services to the community.

Future Developments:

Integration of the ZOO-Kernel into QGIS for accessing the ZOO-Project's processing capabilities from QGIS.

Deliverables

1. Python framework for service execution in ZOO-Project.
2. QGIS processing toolbox made available as First Citizen services of the ZOO-Kernel over the web
3. Documentation for the Python framework and the services added to the ZOO-Project
4. Exhaustive UI demo for using the services added during this project and for adding other QGIS processes as ZOO-Services

Timeline

Proposed timeline for the project:-

Table 3: Project timeline

Time-period	**Tasks and targets**
Community bonding period: May 20th - June 12th	1. Create a wiki page for the project. 2. Get in touch with the community, introduce my project ideas, and incorporate feedback. 3. Update the proposal and share it with the community. 4. Set up the development environment 5. Study the code-base for the QGIS processing framework by fixing a small bug and writing some simple unit tests. 6. Submit a pull request for the bug fix to the official repository.
Coding Phase 1: June 13th - July 20th	1. Implement a Python framework for service execution 2. Test the framework by implementing a simple 'Hello world service 3. Add two QGIS processes as ZOO-Services 4. Document the service execution framework, and added services. Create UI demos for the added services
Evaluation for phase 1: July 21st - July 29th	Phase 1 evaluation
Coding Phase 2: July 25th -September 4th	1. Add six QGIS processes as ZOO-Services 2. Document the added services, and create UI demos for the added services
Evaluation for phase 2: September 5th - September 12th	Phase 2 evaluation
Extended coding phase: September 12th - November 13th	1. Add the final set of twelve QGIS processes as ZOO-services 2. Document the added services and create UI demos for the added services
Final report: November 14th - November 21st	Compile the Final report for the project
Final evaluation: November 2nd - November 28th	Final project evaluation

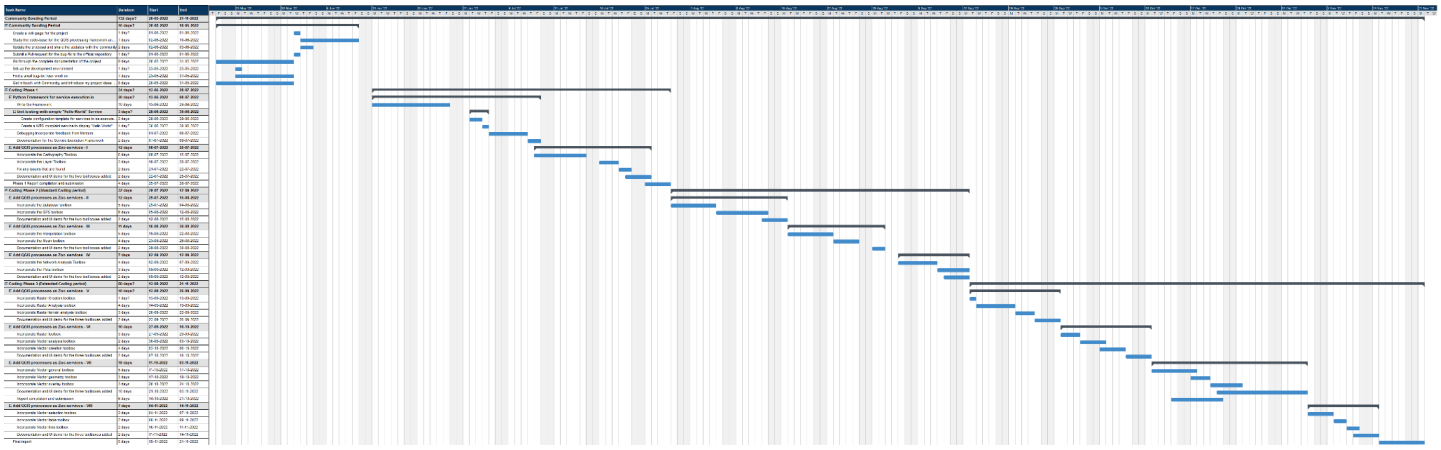


Fig. 3. Detailed Project timeline as a Gantt Chart

(Please find a high-resolution version of the above image here)

Do you understand this is a serious commitment, equivalent to a full-time paid summer internship or summer job?:

Yes, I understand. I'm looking forward to giving SoC my best.

Do you have any known time conflicts during the official coding period?:

No, I do not have any known time conflicts during the official coding period.

Studies

School and Degree:

Table 3: Educational details

Degree	**Institution**	**Time**
M. Tech+PhD (Dual degree) - Geoinformatics	IIT Bombay	2018-Present
B. Tech - Computer Science & Engineering	SRMIST	2013-2017

Would your application contribute to your ongoing studies/degree?

Yes, My application would contribute to my ongoing degree. My research involves the development of machine learning models for mapping minerals on the Martian surface using orbital remote sensing data and demonstrating their use through web-GIS. The skills I gain from this project would help me make setup the server for this work and directly contribute to my degree.

Programming and GIS

Computing experience: Windows, Ubuntu

GIS experience as a user: Esri ArcGIS, ENVI, Erdas Imagine, QGIS

GIS programming and other software programming:

1. Developed an android app for logistics and supply chain management for armed forces which were runner-up at Esri's Mapp Your Way 3.0 challenge. ArcGIS Online was used for it.
2. Developed a GIS and IoT based municipal waste management solution using Javascript and Google's firebase, which won the first prize at the Indian National Academy of Engineer's Hackathon.
3. Developed web apps using OGS standard WMS, WFS, WFS and SOS services
4. Proficient in Python, Javascript, C/C++, and Java

GSoC Participation

Have you participated in GSoC before?

No, I have not.

Have you applied but were not selected? When?

Yes, I applied, in 2021, for the idea 'Augment PostGIS 3.2 with GIST support added to PG14' under the OSGeo umbrella organization

Have you submitted/will you submit another proposal for this year's GSoC to a different org? Which one?

This is the only proposal I will be submitting.

Proficiency test Requirements

1. **Test for the understanding of Github (Version Control):**

   The forked repository can be found here.

2. **Test for the understanding of Web Services:**

   1. **Adding WMS layers:-**

      a. Open the Add WMS layer dialogue box by selecting the appropriate options in the drop-down menus.

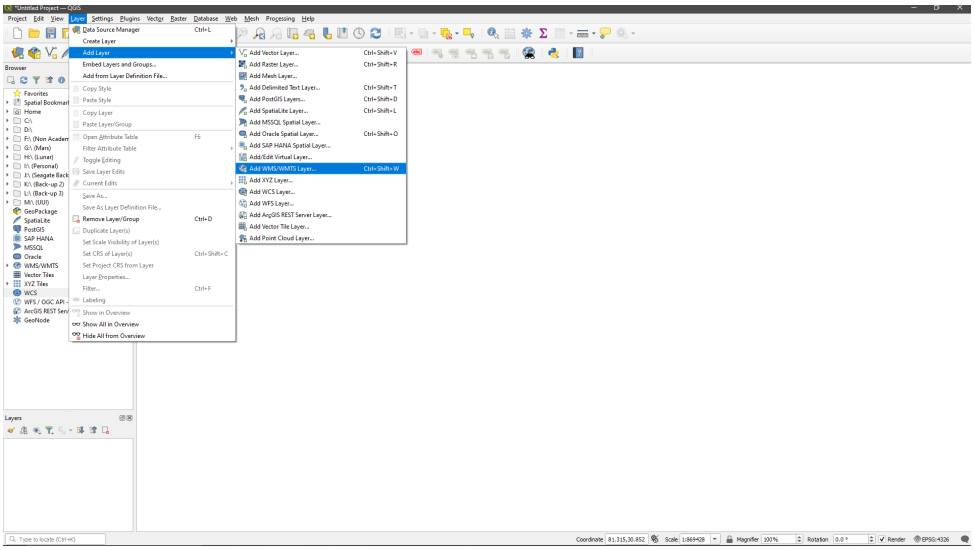


Fig. 4. QGIS drop-down menu option to add a manage WMS connection

      b. Add a new Connection i.e data source. Bhuvan the spatial data interface maintained by ISRO was used here.

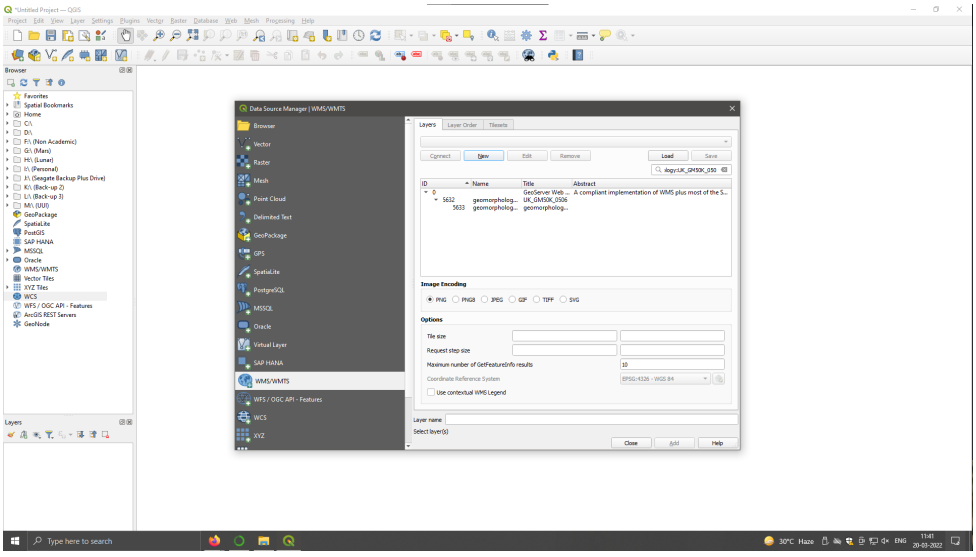


Fig. 5. WMS management dialogue box in QGIS

c. The dialogue box to add the new Connection

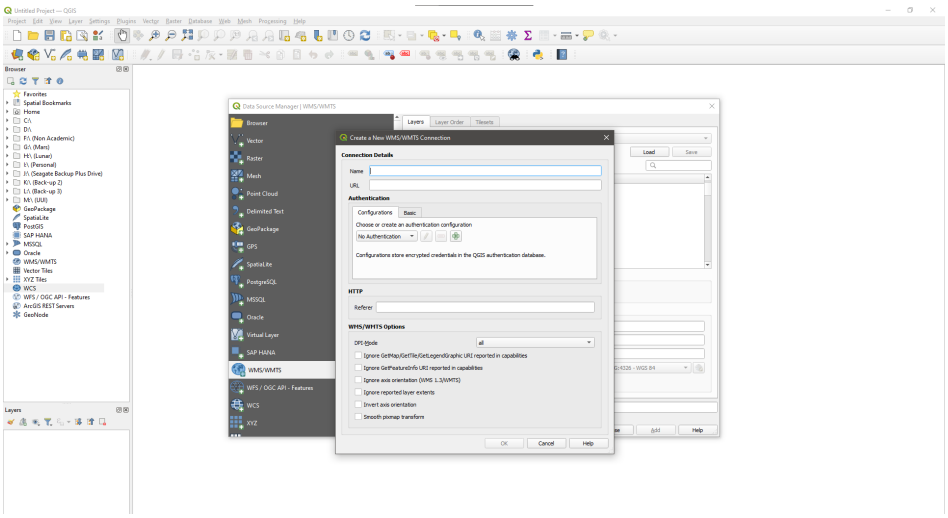
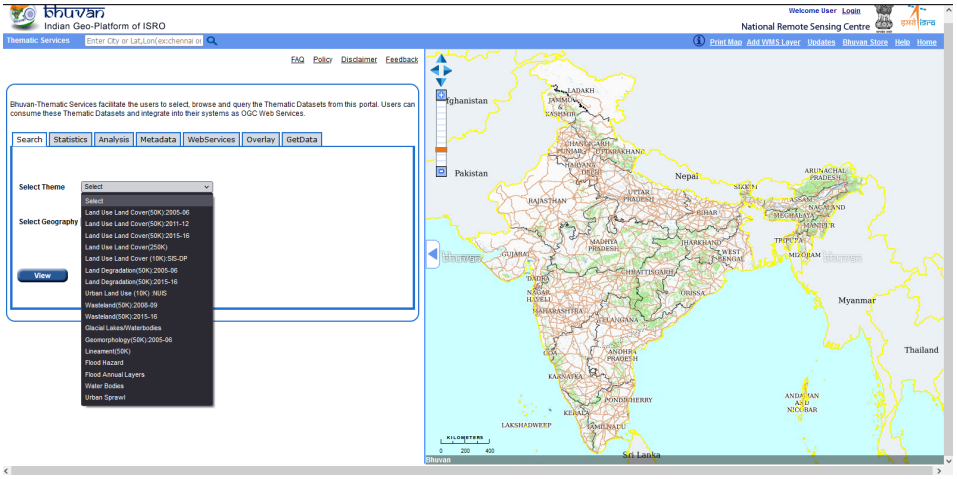


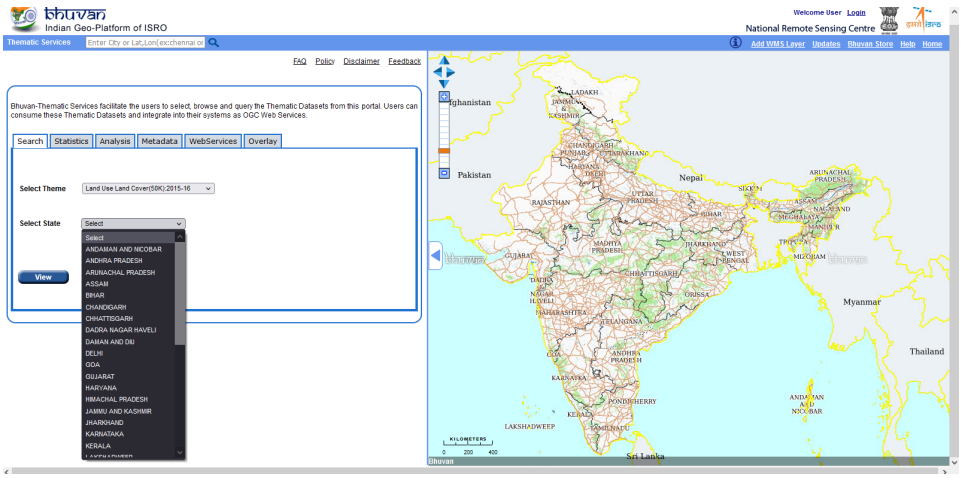
Fig. 6. Dialogue box to add new WMS data-source

d. To get the URL we go to the Bhuvan interface at https://bhuvan-app1.nrsc.gov.in/thematic/thematic/index.php

As a sample we will be working with the LULC map for the Indian state of Maharashtra. For this, we select the LULC map and Maharashtra region form the drop menus



(a)



(b)

Fig. 7. ISRO's Bhuvan interface to get WMS data source

The results for reference can be seen on the same webpage

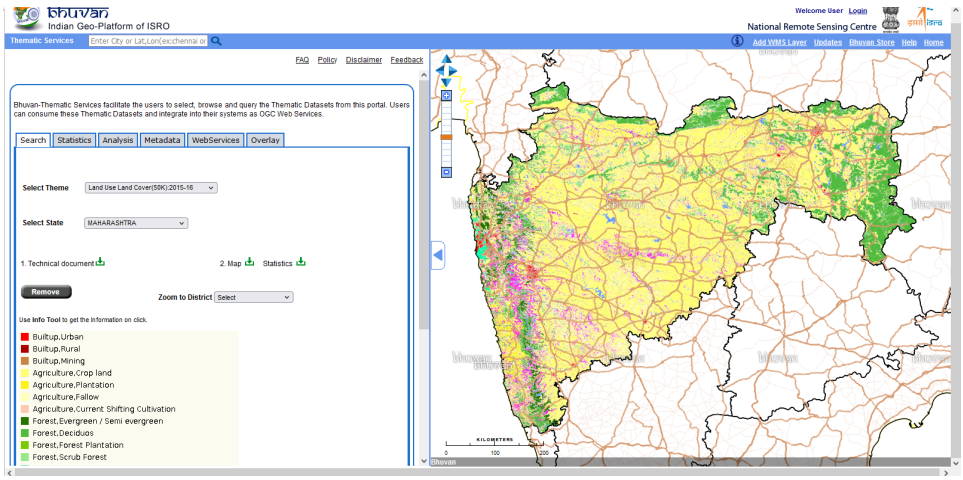


Fig. 8. LULC map of Maharashtra, India on the Bhuvan interface for reference

e. To get the URL and name of the layer we go to the 'WebServices' tab

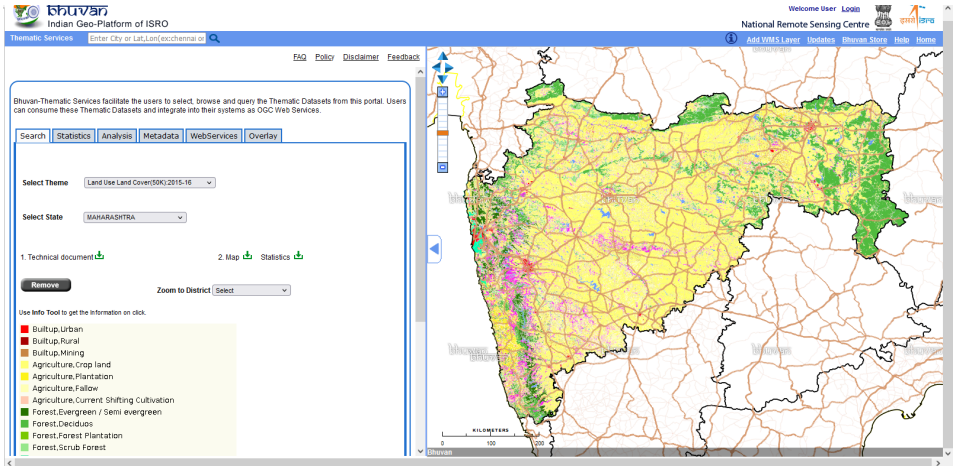


Fig. 9. Bhuvan interface to get the URL for to get the data through a WMS

f. We add the URL to the new connection dialogue box and a name for the connection. As the LULC map was the one of interest. The connection was named, "Bhuvan LULC'

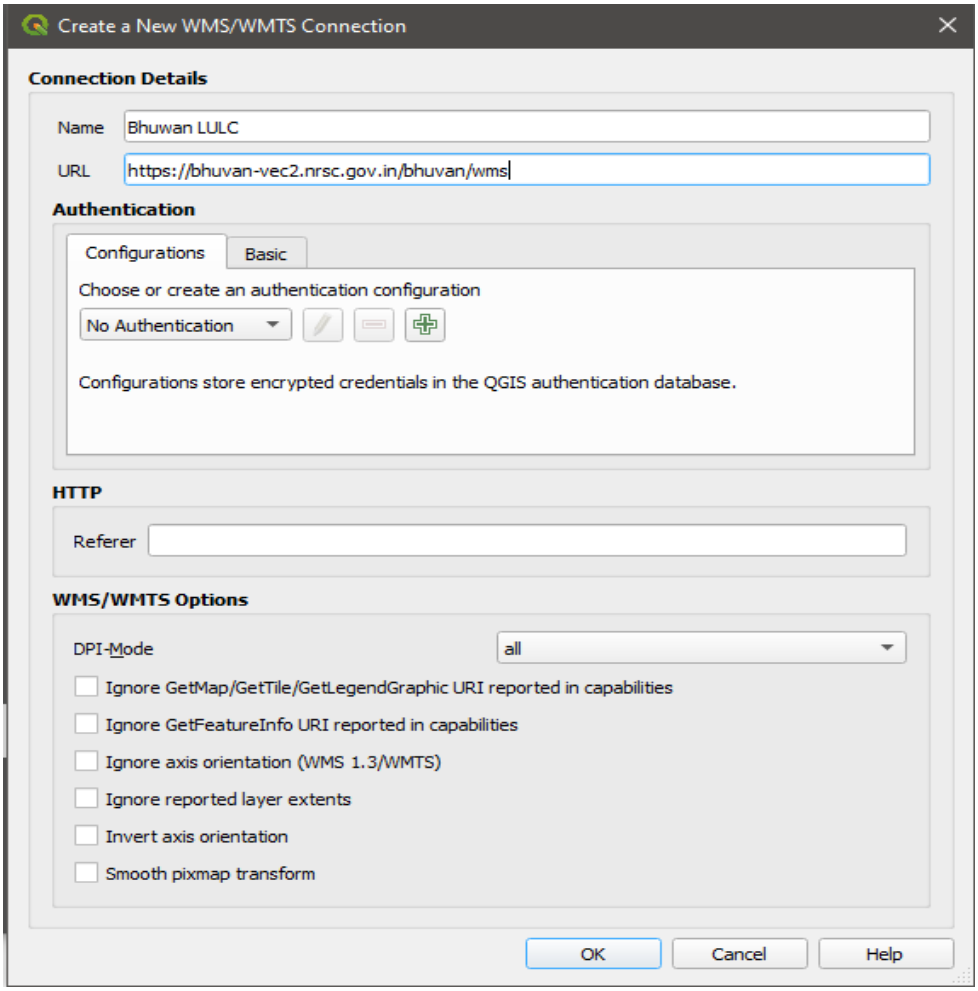


Fig. 10. WMS connection creation window with the Bhuvan URL

g. Once the connection is added. The required layer was selected by searching for its title and then added to the project.

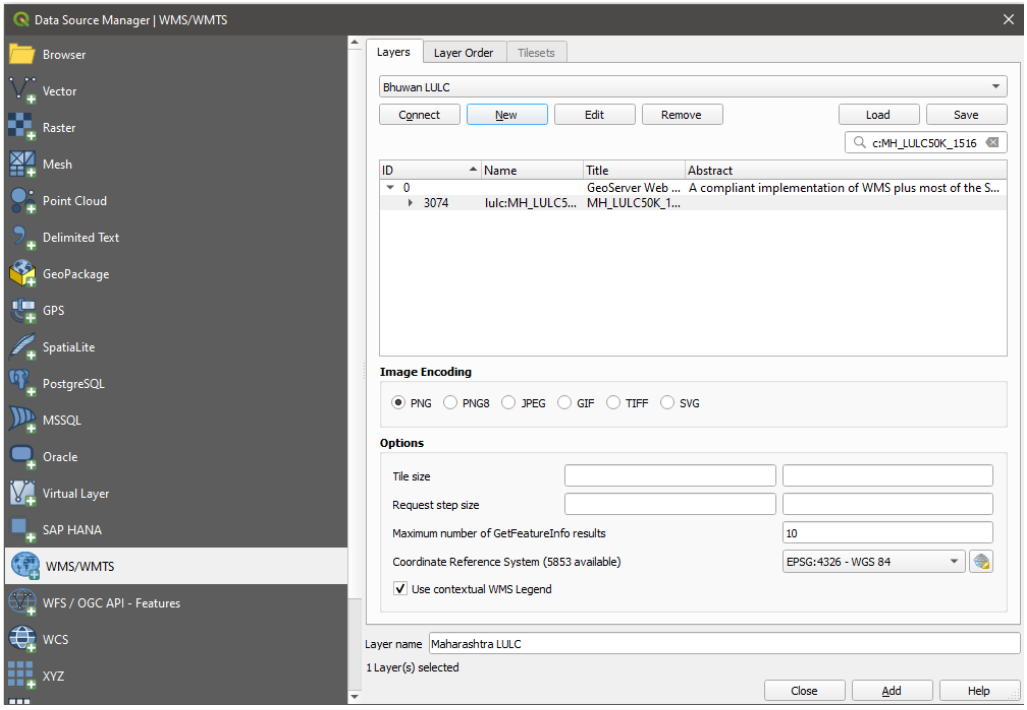


Fig. 11. WMS Data source manager with the WMS connection added and Layer selected

h. The added layer can be seen here.

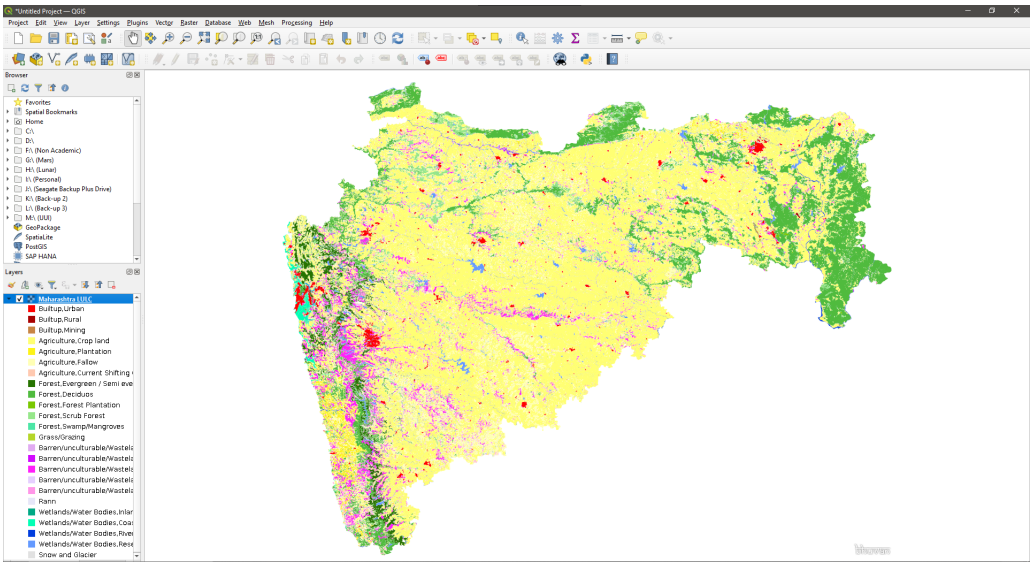


Fig. 12. Maharashtra LULC map rendered in QGIS through the WMS connection

i. Similarly, a WMS layer corresponding to the Geomorphology of Maharashtra was also added.

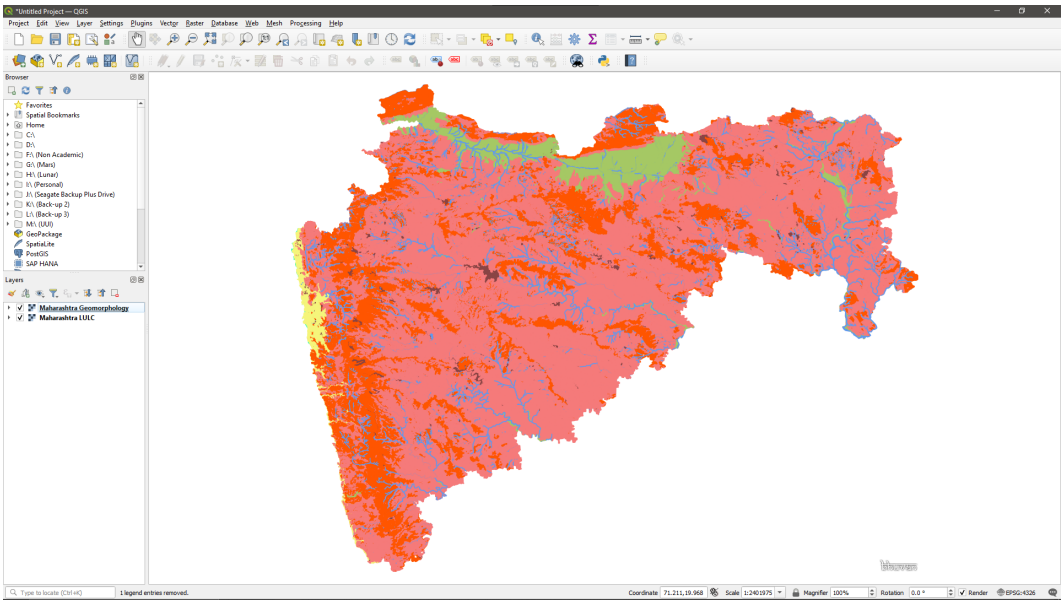


Fig. 13. Maharashtra geomorphology map rendered in QGIS through the WMS connection

2. **<u>Adding WFS layers:-</u>**

a. Open the Add WFS layer dialogue box by selecting the requite options in the drop-down menus shown below.

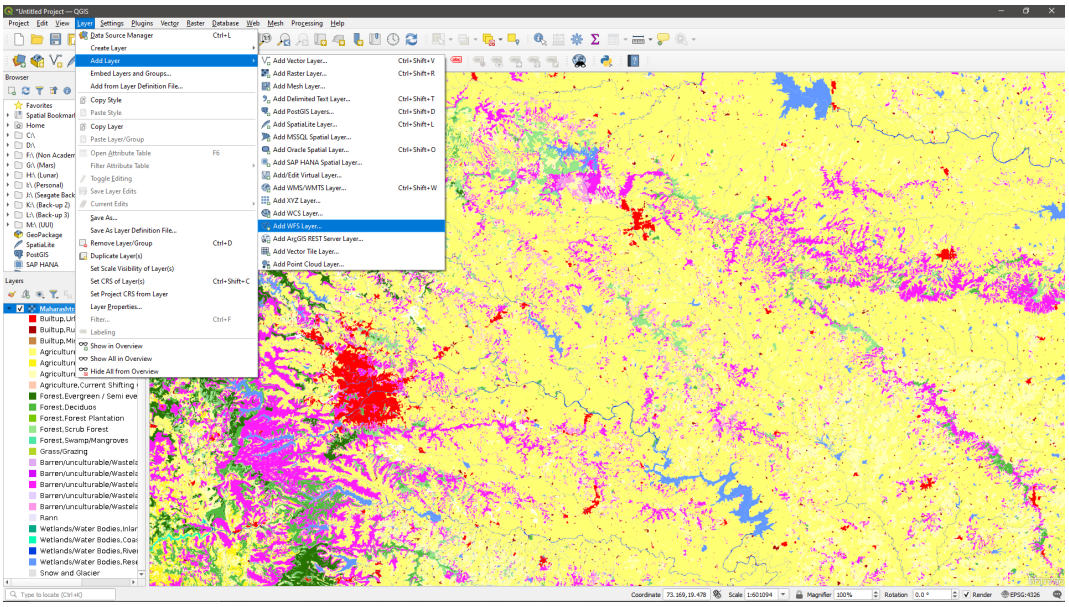
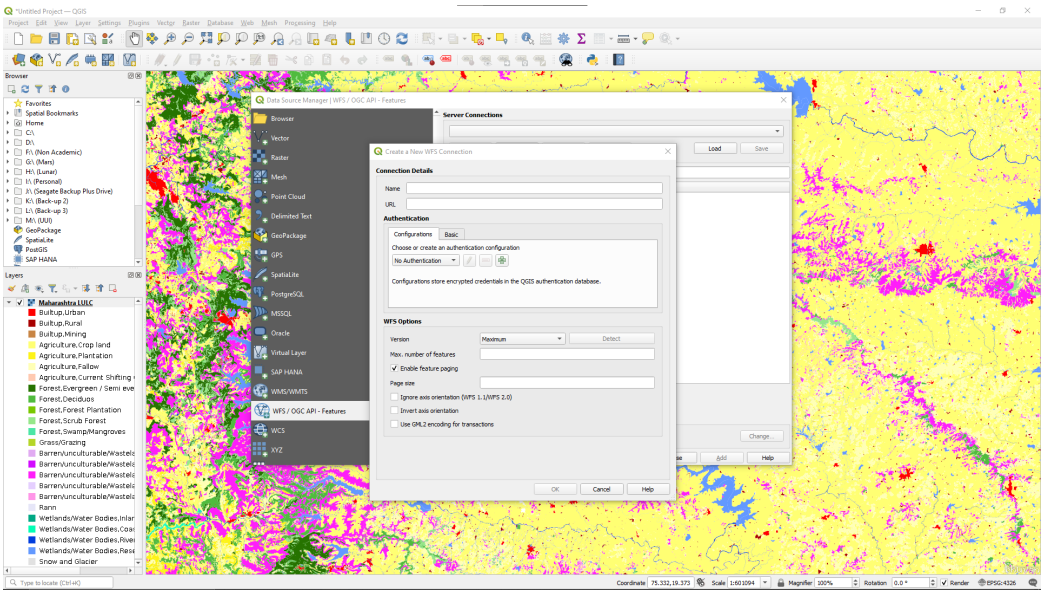


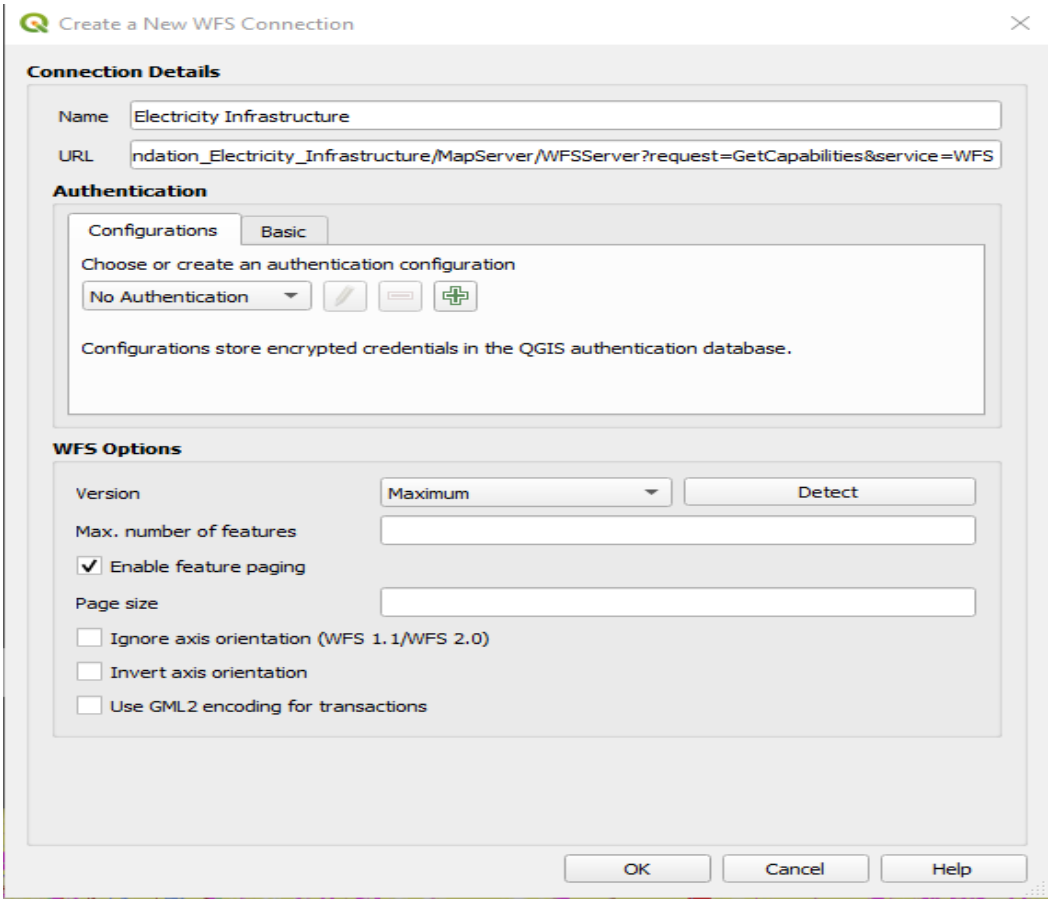
Fig. 14. QGIS drop-down menu option to add a manage WFS connection

b. Similar to WFS, a new connection was added. The tutorial created by Geoscience Australia to add Electrical Infrastructure maps

(https://www.ga.gov.au/data-pubs/Web-Services/using-wms-and-wfs-in-qgis) was followed.



(a)



(b)

Fig. 15. QGIS dialogue box to add a new WFS connection

c. Three available layers were added.

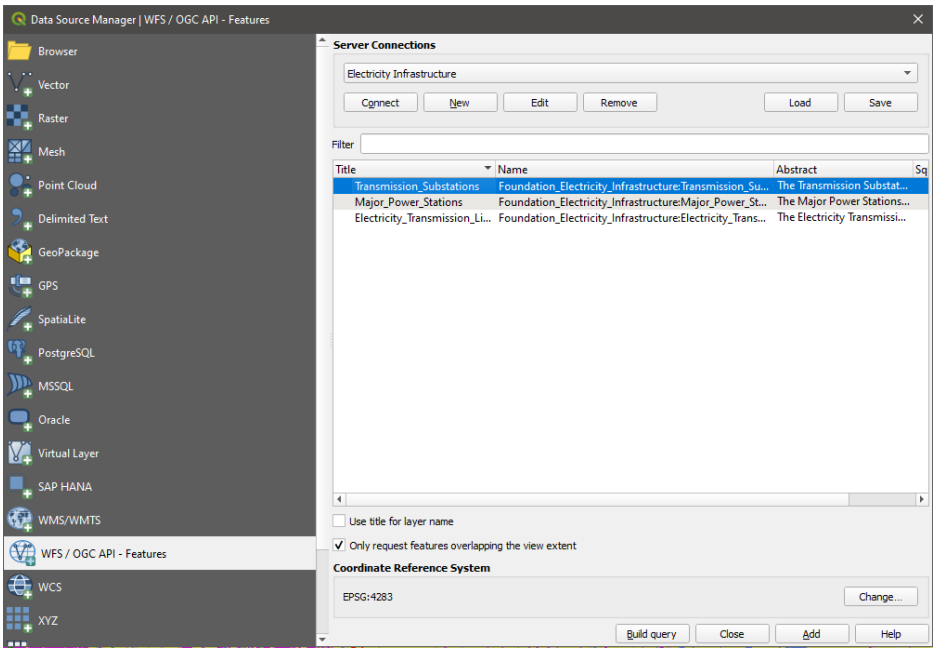


Fig. 16. QGIS dialogue box with three new layers as WFS data sources

d. The added layers are shown in the following figure.

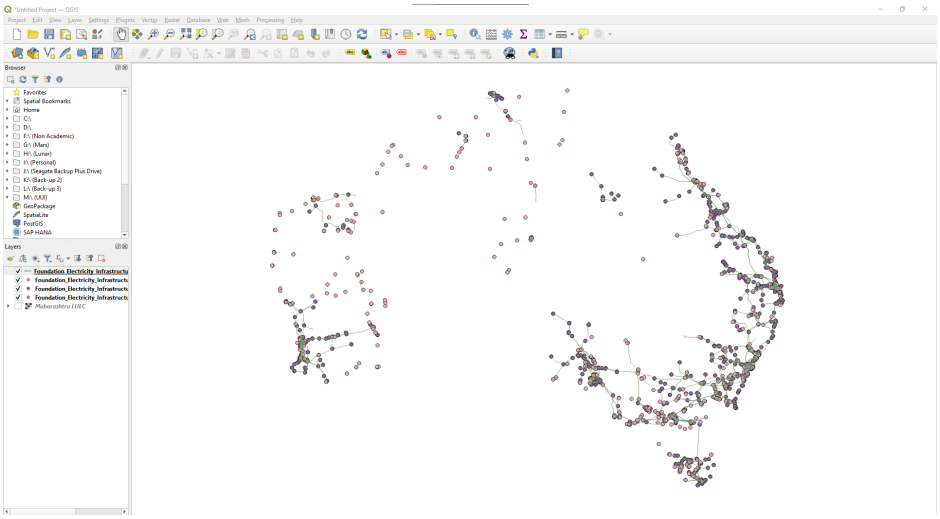


Fig. 17. QGIS with the electricity lines and station in Australia rendered through a WFS connection

3. **Processing using WPS:-**
   Incomplete

3. **Create a pull request to the ZOO-Project repository with your code and report:-**
   Report added and request made.

References

1. The Web Processing Service standard by Open Geospatial Consortium (OGC).
2. The Zoo-Process Documentation.
3. The GDAL documentation.
4. The CGAL documentation.
5. The GRASS GIS documentation.
6. ISRO Bhuvan platform.
7. QGIS processing framework documentation.